

# Fishing for Phishing from the Network Stream

Anirudh Ramachandran<sup>†</sup>, Nick Feamster<sup>†</sup>, Balachander Krishnamurthy\*,  
Oliver Spatscheck\*, Jacobus Van der Merwe\*

<sup>†</sup> Georgia Tech    \* AT&T Labs–Research

## ABSTRACT

Phishing is an increasingly prevalent social-engineering attack that attempts identity theft using spoofed Web pages of legitimate organizations. Unfortunately, current phishing detection methods are neither complete nor responsive because they rely on user reports, and many also require client-side software. Anti-phishing techniques could be more effective if they (1) could detect phishing attacks automatically from the network traffic; (2) could operate without cooperation from end-users. This paper performs a preliminary study to determine the feasibility of detecting phishing attacks in real-time, from the network traffic stream itself. We develop a model to identify the stages where in-network phishing detection is feasible and the data sources that can be analyzed to provide relevant information at each stage. Based on this model, we develop and evaluate a detection method based on features that exist *in the network traffic itself* and are correlated with confirmed phishing attacks.

## 1. Introduction

Many schemes to detect and prevent phishing have been proposed over the past few years [1, 2, 6, 7, 11, 13, 18, 19, 22, 23, 25], yet losses for victim organizations and users due to phishing continue to mount [10]. Today, large-scale phishing prevention primarily relies on constructing “blacklists” of URLs from user reports (e.g., the Google Blacklist [11]). This reactive approach makes timely response to every attack challenging, if not impossible: before a phishing URL is reported, verified, and listed, many users may have already fallen prey to the attack.

Instead, we propose a complementary, *proactive* approach called *Fish4Phish* that detects phishing attacks from the network traffic itself. Our approach involves monitoring a variety of network derived information sources, e.g., email traffic, DNS information, the HTTP clickstream, and statistics about traffic flows, and *correlate these observations* to detect phishing attacks. This approach has several advantages over existing schemes. First, it does not rely on user cooperation: Once a phishing attack is identified, all users can be protected against it, irrespective of their client software. Second, in-network mitigation can be applied anywhere along the network path. Third, by correlating multiple sources of information that are readily available inside the network, in-

Contact emails: {avr, feamster}@cc.gatech.edu  
{bala, spatsch, kobus}@research.att.com

Georgia Tech CSS Technical Report GT-CS-08-08

network monitoring can potentially detect phishing attacks more quickly and more completely than existing techniques.

*Fish4Phish*’s detection features and heuristics are based on a multi-stage model of phishing attacks that (i) help operators reason about when, where, and how the attack might be detected and (ii) identify various datasets associated with each of the stages. *Fish4Phish* uses these features to determine the likelihood that a particular URL corresponds to a phishing attack. The model that serves as the basis for *Fish4Phish* suggests *all* possible opportunities for detection; an organization can tap any subset of these opportunities depending on what data might be accessible or available.

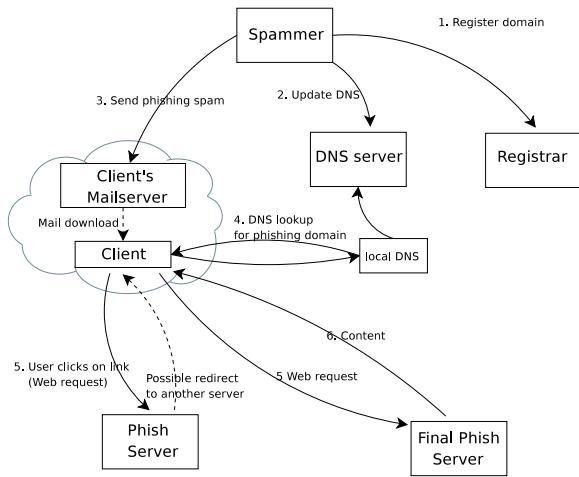
In this paper, we evaluate *Fish4Phish* on a dataset of clickstream URLs; consequently, we use two methods that are applicable to this data. The primary technique is to detect phishing URLs using automatically generated regular expressions. To reduce false positives from the regular expression matcher, we use an auxiliary heuristic: classifying Web servers using their estimated activity periods. We make the following contributions:

- We present a model for phishing attacks that serves as the basis for determining data sources useful for detecting phishing. The model also suggests features that can be applied to any network that collects (or can collect) the corresponding data.
- We evaluate *Fish4Phish* using data from a large access network that provides Internet connectivity to approximately 20,000 homes. Our results show that: (i) many users are clicking through on phishing URLs, and (ii) a few features can be combined to provide good detection with low false positive rate.

## 2. Related Work

Existing phishing detection systems fall into 3 classes: (1) *client-side solutions*; (2) *client-server solutions*; and (3) *server-side solutions*. We briefly survey samples of previous work in each class to show how *Fish4Phish* differs from these efforts; we also briefly discuss model and feature-based phishing detection methods.

**Client-side solutions** are the most common among anti-phishing tactics. Many of these solutions are available as Web browser “plugins”. *Spoofstick* prominently alerts the user about the current domain hosting a URL [22]. *Spoofguard* performs various checks on the URL and content of a Web page and raises alarms on suspicious web accesses [7]. *PwdHash* modifies the password that is submitted to a site as a function of the site’s domain name so that the password



**Figure 1: Model of a phishing attack. Each step has a possible detection opportunity.**

submitted to a phishing site is not the same as the one submitted to the legitimate site [6]. *Fish4Phish* performs similar checks to these tools but also incorporates additional information from the traffic stream to reduce false positives.

**Client-server solutions** such as Google Safe-Browsing Extension [11], Netcraft Toolbar [18], Phishtank Toolbar [19] and “blackbox” security appliances sold by vendors that filter URLs [5, 21, 24] query blacklists to identify phishing URLs. iTrustPage [14] uses a search engine to retrieve reputation information about URLs. Although many blacklist-based schemes have high accuracy rates, they may be slow to list some URLs [8].

**Server-side solutions** such as Sitekey (a visual cue system present on Bank of America’s sites [2]) try to increase user awareness. Such methods often fail as users are less likely to notice the *absence* of a visual indicator than its presence.

**Modeling and feature-based classification of phishing** Previous work has attempted graphical modeling of phishing attacks using actions and probabilities [15]; in contrast, our model reflects macro properties for phishing that are visible in the wide-area. We study features that have been motivated in previous work. SpoofGuard uses regular expression matching [7], and others have observed that phishing sites possess low uptimes [3, 4, 17]. Finally, previous work has suggested looking for low DNS TTLs to detect phishing servers using fast flux [17].

### 3. A Model for Phishing Attacks

In this section, we describe a model for phishing and enumerate the possible data sources that could be used to detect phishing in the network. Based on this model, network operators can develop techniques for detecting phishing attacks from various types of network data available to them.

Figure 1 shows the sequence of steps involved in a phishing attack. Table 1 summarizes these detection opportunities, possible indicators of phishing attacks at each step, and the availability of each of these datasets. We describe the steps in our model below.

**Step 1: Domain registration.** Most phishing sites are

hosted on newly registered domains. Domain registration zone file logs can expose suspicious patterns of registering and unregistering domains. Moreover, domain names can be tested for similarity to phishing domain names using regular expressions. Recent measurement studies have indicated that it is indeed possible to discover phishing domain registrations and de-registrations using this source [16].

**Step 2: DNS update for phishing domain.** Recent reports indicate that phishing sites exhibit “fast flux” behavior, where the mappings of both names to IP addresses and names to authoritative nameservers are continually changing. Monitoring DNS records for a domain over time could also allow a detection system to determine whether a particular domain was likely being used for a phishing attack. A detection algorithm could also query the DNS to look for authoritative nameservers hosted on broadband or dialup machines (or on other short-lived connections).

**Step 3: Phishing email.** The next step in the phishing process is “fishing” for vulnerable users by sending out a phishing email (or messages on web-based message boards and instant messages [9, 20]), with a URL that directs users to the newly registered domain. Emails that contain phishing URLs could be detected either at the mail server or at an appliance that examines network traffic. An advantage to detecting phishing attempts at this stage is that phishing URLs in emails can be *proactively* detected, before the attack even reaches a user’s inbox.

**Step 4: DNS lookup for the phishing domain.** After a phishing email reaches a user’s inbox, the user may click on a URL contained in that email. If the URL involves a DNS lookup, monitoring these lookups provides another opportunity to detect phishing attacks. At this stage, the monitor could then look for evidence of fast-flux behavior [12], including low time-to-live (TTL) values for DNS A or NS records, or changes to the records themselves [17].

**Step 5: HTTP request for phishing Web site.** A detection mechanism could examine several features that might indicate that a client was making a request to a suspect server. First, recent studies suggest that HTTP scam and phishing sites are typically short-lived [3, 17, 4], which suggests that HTTP requests to Web servers with short uptimes may be cause for suspicion. Second, the HTTP requests themselves may provide an indication that a user is making a request for a URL that is a likely phishing attack, because many phishing URLs disguise the path (*i.e.*, GET) portion of the URL to be similar to a target organization’s URL. Third, looking for HTTP requests to servers that are on IP blacklists, known bots or compromised hosts, or otherwise “suspect” IP addresses might also prove to be helpful indicators. Finally, request patterns such as the presence of HTTP redirects may indicate an attempt to send a client to an ephemeral site (*e.g.*, a phishing site hosted by a botnet).

**Step 6: HTTP response (including content).** Once the request is issued and the client follows a sequence of redirects, a Web server returns the content to the client. This content

Step	Description	Hypothesis	Data	Location
1	<i>Domain registration:</i> Phisher registers domain for hosting phishing site.	(1) Domain names match a regular expression that is common to phishers; (2) Domain name registration / deregistration shows large flux	Domain registration logs	Registrar
2	<i>DNS update for phishing domain:</i> Phisher updates DNS NS and A records.	(1) DNS records may match a regular expression; (2) TTLs for DNS records may be short; (3) DNS servers IP addresses may cluster towards certain prefixes.	DNS responses collected from proactive queries of URLs in spam.	Can be collected with active measurements
3	<i>Phishing email:</i> Phisher sends email with content of phishing attack (including URL).	Email contains URL whose regular expression matches a regular expression	Emails from spam traps and user mailboxes	Local network
4	<i>DNS lookup for phishing domain:</i> A user may request a phishing URL in an email, etc., generating a DNS lookup.	Same as 2 above	Passively collected DNS queries and records	Local network
5	<i>HTTP request for phishing Web site:</i> A user's click will generate an HTTP request for the phishing Web site.	(1) The HTTP request may contain domains that match regular expressions; (2) the HTTP request may involve redirects	HTTP traffic "on the wire"	Local network
6	<i>HTTP response (and content):</i> Client receives content from phishing Web site.	Aspects of content may resemble content from the corresponding legitimate Web site	Reassembled Web pages from packet traces	Local network

**Table 1: Steps in a phishing attack, data available for monitoring at each step, and phishing indicative behavior in each step.**

is also visible in the network stream. In some cases, sophisticated analysis may be able to reconstruct the returned content and compare it to that of a legitimate Web site. An alternative method would be to look at how the page is *constructed*: for example, a phishing site might construct a page by assembling remotely hosted content (possibly even from the legitimate site itself).

## 4. Fish4Phish Features

Using the model of phishing from Section 3 as a guide, this section evaluates the possibility of using three specific features—URL structure, Web server activity period, and DNS TTL values—corresponding to Steps 4 and 5 in the model. Ultimately, an on-the-wire phishing detection system could use other features; we focus on these three features because the data was readily available.

### 4.1 URL Patterns and Structure

Phishing URLs attempt to deceive users by constructing URLs that appear similar to the URL of the phishing victim. To better understand URL structure, we analyzed a sample of 1000 phishing URLs from the Google blacklist from June 2007–September 2007 and found that 86% of URLs exhibited one of the following two patterns<sup>1</sup>:

1. *Victim's URL prepended to phishing URL domain.* (e.g., `http://signin.ebay.com.910462.phishsite.com/sc/saw-cgi/eBayISAPI.dll/index.php`). 573 out of the 858 identifiable URLs had this structure.
2. *Victim's URL appears in the GET string.* (e.g., `http://phishsite.com/signin.ebay.com/ws/eBayISAPI.php`). The remaining 285 phishing URLs had this structure.

The closer a URL appears to the target organization's URL, the more likely it is a phishing URL. The observation that most phishing URLs conform to one of these two

<sup>1</sup>The remaining 14% of URLs were typically in one of the following categories: (i) no useful information (e.g., used an IP address), (ii) multiple victim names in URL, or (iii) no identifiable victim name in URL.

basic structures allows us to construct a regular expression matcher that detects almost all these URLs with few regular expressions. Section 5 describes the design of this regular expression matcher in detail.

### 4.2 Web Server Activity Period

Many Web servers that host phishing sites are short-lived: a recent study showed that phishing sites are accessible for 3.8 days on average, and these sites are never active for longer than 30 days [3, 4]. The short activity periods of these Web servers likely reflect attempts to remain undetectable and untraceable, or to evade blacklists. Accordingly, given a likely phishing URL from the regular expression matcher, we can eliminate false positives if the server hosting the URL has been active for longer than some fixed period of time. Section 5 describes this process in more detail.

### 4.3 TTLs of DNS records

Phishing servers may be hosted on compromised machines that are often hidden using advanced DNS tricks such as fast-flux (or double-flux) [12, 17]. Although many legitimate sites may set low TTLs, we expected that phishing sites would typically set low DNS TTLs. To test this hypothesis, we set up a local mail delivery agent at a spam trap to automatically pass the headers and body of each new message to a program that extracts all URLs in the message and issue a DNS ANY query for each domain name. The ANY query returns the A, NS, and MX (if available) records.

## 5. Fish4Phish

**Step 1: Whitelisting.** *Fish4Phish* applies a two-phase regular-expression matching algorithm to detect phishing URLs. The *whitelist matcher* takes as input all URLs and discards those URLs that have legitimate domain names of the organizations that *Fish4Phish* tries to protect. The *whitelist matcher* also includes popular domains such as `www.google.com` in order to avoid unnecessary computation for the blacklist matcher (next step). Because we cannot include every legitimate domain in the whitelist,

*Fish4Phish* whitelists domains that either receive a substantial amount of traffic or commonly appear in the list of false positives after blacklist matching, based on feedback.

**Step 2: Blacklisting.** From the remaining set of non-whitelisted URLs, the *blacklist matcher* searches for all URLs similar to target organizations’ URLs. The blacklist matcher checks both parts of a URL—the domain name and the HTTP GET/POST string—for similarity with each target organization’s URL. The blacklist matcher gives a higher score to a URL that matches more clauses in the regular expression. For example, `(signin|cgi|secure)?(.*?) (ebay) (.*?) (com)?.+([[:digit:]]+)?` is a simplified version of the blacklist expression that looks for Ebay phishing in the domain name of a URL. `[[:digit:]]` represents the character class containing digits. Two URLs that match this regular expression are `ebay.doubleclick.net` and `signin.ebay.com.phishsite.com`. The second URL matches more clauses in the regular expression and will receive a higher score, even if we know nothing about the legitimacy of either domain. A URL which matches no clause receives a zero score.

**Step 3: Whitelisting feedback.** Given only an initial seed whitelist, the blacklist matcher generates many false positives. Common examples of such false positives are advertisement domains that use keywords of target organizations to personalize their ads, e.g., `http://ebay.doubleclick.net` may trigger a regular expression that looks for `http://ebay.phishsite.com`. We periodically update the *Fish4Phish* whitelist to reduce the number of false positives that result from the blacklist. To reduce false positives, *Fish4Phish* incorporates feedback from users or operators about false positive URLs. These reports are incorporated into the whitelist. We show in Section 6.2 that this mechanism for tuning the whitelist dramatically reduces the false positive rate over time.

**Step 4: Reducing false positives based on server activity.** *Fish4Phish* uses the activity period for a Web server to reduce false positives. The first step in determining a server’s activity period is to *map the potential phishing URL to a corresponding server’s IP address*. When HTTP requests can be directly monitored, obtaining this mapping is trivial: the packets on the wire containing the GET request contain the destination IP address of the server. Given the IP address of the Web server corresponding to a URL, the second step involves *estimating the time interval over which the server was active*. *Fish4Phish* relies on passive monitoring of TCP flow to determine the length of time a server has served connections from clients. This measurement provides a *lower bound* on the time interval for which the server has been active (i.e., we can say that the server has been active for at least that period of time). Using this lower bound, *Fish4Phish* eliminates potential false positives by removing servers from suspicion if they have been active for a period much longer than typical phishing server uptimes (typically a few days).

	URLs	Domains	Servers	Clients
<b>Verified Phishing</b>	227	34	26	64
<b>False Positives</b>	20,668	831	4,589	6,214

Table 2: URLs matched by the URL structure feature.

For each IP address, *Fish4Phish* maintains a start and end time (i.e., the time a request was first seen to that IP address, and the time it was last seen). In practice, both TCP flow logs and traffic flow statistics are sampled. The effects of this sampling, however, will only cause *Fish4Phish* to underestimate a server’s activity period, making the server less likely to be thrown out as a false positive.

## 6. Evaluation

We evaluate the features from Section 4 using HTTP clickstream data from an access network, and DNS data from a large campus network. We investigate whether: (1) Our features can detect more phishing URLs than publicly available sources such Google Blacklist; (2) The features detect phishing URLs faster than Google Blacklist; (3) Combinations of features reduce false positive results. We also evaluated a third feature—whether DNS TTLs might be useful for distinguishing phishing domains from legitimate ones. Although we do not present the results here due to space constraints, we found, contrary to both our expectation and suggestions from recent previous work [17], that low DNS TTL values do not appear to be useful for distinguishing phishing URLs from legitimate ones.

### 6.1 Data

We collect HTTP clickstream data from an access network of an ISP, representing approximately 20,000 home user accounts, from October 21–28, 2007. Each line of the data contains one HTTP header (e.g., ‘Host’, ‘GET’, ‘Referer’, ‘User-agent’, etc.), a high-resolution timestamp, and a 4-tuple comprising the source and destination IP addresses and ports.

Because of bandwidth limitations at the data collection node, we were unable to *independently* look for the two relevant features: the URL structure feature and activity period feature. Instead, we apply only the URL pattern matching on the clickstream data at the collection node, and record each URL (and its corresponding flow 4-tuple) that received any non-negative score (and not necessarily a high score). We perform this step primarily to reduce the volume of data so that the resulting logs can be further analyzed. Over the 8-day analysis period, the URL structure classifier flagged 20,895 URLs from 865 unique domain names<sup>2</sup>.

To estimate server activity periods, we examine TCP flow logs and produce a hourly summary for *every* destination IP that contains the time, mean inter-arrival time, standard deviation of inter-arrival time distribution, and number of

<sup>2</sup>The domain name for a URL is obtained from the ‘Host: ’ HTTP header field. We examine only the part of the domain that uniquely identifies an organization, which is usually the last two or last three portions of the dot-separated domain name. For example, `www1.bbc.co.uk` and `www2.bbc.co.uk` have different absolute domain names, but the portion that identifies the organization, `bbc.co.uk`, is the same for both. A domain name comprising only an IP address is used as is.

Day	Phishing	Legitimate		
	phish (Detections)	FPs (Alerts)	# WL	FP Rate
1	7	131	1857	.071
2	12	143	2208	.065
3	11	117	2733	.043
4	26	121	2981	.041
5	3	88	1957	.045
6	39	66	3771	.018
7	13	74	2815	.026
8	116	91	1515	.060
Totals	227	831	19,837	

Table 3: Statistics of alerts raised each day. “phish” indicates alerts that correspond to detected phishing attempts; there were no missed detections. FPs are URL alerts that were subsequently whitelisted. WL shows the URLs that were automatically whitelisted without ever generating an alert.

unique remote IPs that connected to it. We join this data with the IPs of servers hosting URLs that were flagged using the regular expression matcher.

Ideally, we would have estimates of each Web server’s activity period *before* a URL corresponding to that server appears in the clickstream (in practice, the classifier would operate on history of the server’s uptime). Unfortunately, we possess TCP flow logs only for the same time period as the clickstream logs, so we cannot see a server’s history prior to a click. Instead, we use the flow information to see whether the activity periods of phishing servers are significantly lower than activity periods of servers hosting false positive URLs.

## 6.2 Results

**Summary.** *Fish4Phish* detected 227 unique phishing URLs from 34 distinct domains, hosted on 26 distinct servers. Table 2 shows, for both phishing and non-phishing URLs, the number of URLs detected, unique domains for these URLs, distinct server IPs hosting these URLs, and distinct client IPs that accessed these URLs.

**URL pattern matcher.** *Fish4Phish*’s URL pattern matching mechanism dynamically updates its whitelist based on reports of false positives (*i.e.*, by a network operator, or based on user reports). We emulate this behavior as follows. We begin with an empty domain name whitelist, proceed over the URL log files chronologically, and as the system raises alerts for URLs that are false positives, we add the domain name to the whitelist (so all further URLs under that domain will be automatically whitelisted). At the end of each day’s log, we count the number of URLs that were (1) verified as phishing, (2) automatically whitelisted, or (3) generated false positives given the state of the dynamic whitelist at that time.

Table 3 illustrates these statistics for the first 8 days of our trace. Our analysis shows that steadily whitelisting domains reduces potential for false positives when those domains reappear. To confirm the trend of decreasing false positive rates, we performed the same analysis on a longer period of 35 days, from October 21–November 25, 2007. Figure 2 plots the false positive rates across the longer trace; false positive rates steadily decrease from 7% false positives to

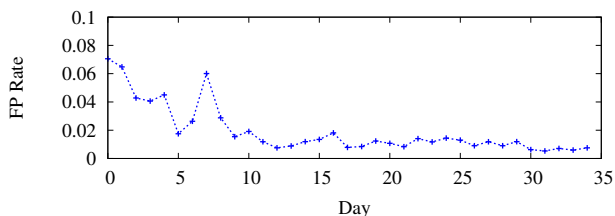


Figure 2: The false positive rate for phishing alerts decreases as the whitelists are updated to include more false positive domains.

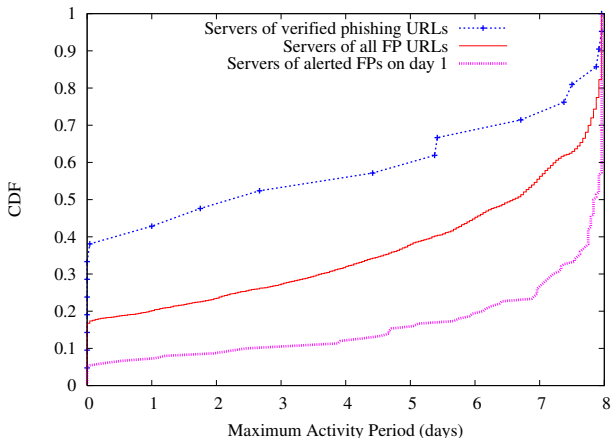


Figure 3: Activity periods for servers that hosted a phishing URL, a false positive URL *before* dynamic whitelisting, and a false positive URL (on the first day) *after* whitelisting.

0.7% within just over a month. There is a jump in false positive rate on day 8 because false positive rate is computed as  $(\# \text{ of } FPs) / (\# \text{ of } \textit{whitelisted domains})$ , and the number of whitelisted domains on day 8 was also low for an unknown reason. This discrepancy indicates that, if entirely new domains appear, the number of false positives may stay steady or even grow. The whitelisting approach does, however, appear to be critical: without user feedback (*i.e.*, given only a static whitelist), false positive rates can be as high as 96%; with user feedback, the number of false positives is much lower.

Providing feedback to the whitelist appears cumbersome, but three aspects suggest that this approach may be manageable. First, the number of URLs that required manual classification (*i.e.*, phishing URLs + false positive URLs) is low compared to the number of URLs that were automatically classified with the whitelist. Second, this classification might be done by users who report this relatively small number of false positives (similar to how users today have toolbar extensions to report email spam missed by filters). Finally, the false positive rate for URLs (*i.e.*, number of false positives per number of negative instances) steadily decreases: as the dynamic whitelist adds more false positive domains, more URLs that would have required human intervention with only a static whitelist can be skipped automatically.

**Reducing false positives.** We compare the activity periods of Web servers that we have verified as phishing sites to: (1) servers corresponding to false positives (*i.e.*, legitimate Web sites), which allows us to compare how well this

feature performs in isolation; and (2) only the set of false positive URLs that were flagged by the first feature, which allows us to evaluate the extent to which this feature might perform for reducing the false positives that were flagged by the URL structure classifier. For the latter case, we use only the first day of false positive URLs, since all of the false positive URLs observed on the first day that escaped whitelisting could have feasibly been active for the remainder of trace period (*i.e.*, as long as 7 additional days).

Figure 3 shows that the average activity period for phishing servers is lower than activity periods of all servers hosting false positive URLs (*i.e.*, legitimate Web servers): even without applying URL matching, setting a *maximum* threshold for activity period for phishing servers to 5.5 days eliminates more than 60% of false positive URLs while retaining almost 70% of phishing URLs. Using the activity-period threshold-based classification *after* applying dynamic whitelisting is more effective: the same threshold of 5.5 days eliminates 85% of false positives.<sup>3</sup>

**Comparison with Google blacklist** To determine the completeness and responsiveness of *Fish4Phish* relative to the Google blacklist, we compared the phishing URLs found by *Fish4Phish* with logs of the Google Anti-Phishing blacklist from October 21–31, 2007. Of the 34 unique phishing domains that *Fish4Phish* found during this period, only 7 domains were listed on Google’s blacklist, all of which were listed on October 31 or later: *Fish4Phish* detected *all* of these URLs at least 2 days before the Google blacklist.

## 7. Conclusion

This paper has presented *Fish4Phish*, a set of techniques for detecting phishing attacks in network traffic. In contrast to existing techniques, which rely primarily on user reports and manual verification, *Fish4Phish*’s detection is automatic: it monitors network traffic (*e.g.*, HTTP clickstreams, spam traps) to detect phishing attacks before they become listed in commonly used blacklists. *Fish4Phish*’s detection heuristics are based on a model of phishing attacks that identifies opportunities for network operators to detect phishing attacks using a combination of various datasets. Our model of phishing presents heuristics and features that may prove useful for detecting phishing attacks from network traffic using the methods we have presented or possibly a more sophisticated classifier. *Fish4Phish* uses dynamic regular expression matching of URLs and a combination heuristics to reduce false positives.

## REFERENCES

- [1] Anti-Phishing Working Group.  
<http://www.antiphishing.org/>.
- [2] Bank of America SiteKey.  
<http://www.bankofamerica.com/privacy/sitekey/>.

- [3] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker. Spamsscatter: Characterizing Internet Scam Hosting Infrastructure. In *Proc. 16th USENIX Security Symposium*, Boston, MA, Aug. 2007.
- [4] Anti Phishing Working Group. Phishing Activity Trends Report. [http://www.antiphishing.org/reports/apwg\\_report\\_june\\_2007.pdf](http://www.antiphishing.org/reports/apwg_report_june_2007.pdf), 2007.
- [5] Barracuda Web Filter. <http://www.barracudanetworks.com/ns/products/web-filter-features.php>.
- [6] D. Boneh, C. Jackson, J. Mitchell, N. Miyake, and B. Ross. PwdHash. <http://crypto.stanford.edu/PwdHash/>.
- [7] D. Boneh, J. Mitchell, R. Ledesma, N. Chou, and Y. Teraguchi. SpoofGuard. <http://crypto.stanford.edu/SpoofGuard/>.
- [8] L. Cranor, S. Egelman, J. Hong, and Y. Zhang. Phishing Phish: An Evaluation of Anti-Phishing Toolbars. Technical Report CMU-CyLab-06-018, CMU, Nov. 2006.
- [9] eweek.com. Phishing Dips into Yahoo IM. <http://www.eweek.com/article2/0,1759,1779798,00.asp,2005>.
- [10] Gartner Report. Consumers to lose \$2.8B to phishers in 2006. <http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9004926>.
- [11] Google, Inc. Google Safe Browsing Blacklist. <http://sb.google.com/safebrowsing/update?version=goog-black-url:1:-1>.
- [12] HoneyNet Project & Research Alliance. Know Your Enemy: Fast-Flux Service Networks. <http://www.honeynet.org/papers/ff/fast-flux.html>, 2007.
- [13] Internet Identity Phishing Detection Service. <http://www.internetidentity.com/>.
- [14] iTrustPage: Pretty Good Phishing Protection. <http://www.cs.toronto.edu/~ronda/itrustpage/>.
- [15] M. Jakobsson. Modeling and Preventing Phishing Attacks. In *Conference on Financial Cryptography and Data Security, Phishing Panel*, 2005.
- [16] D. K. McGrath and M. Gupta. Behind Phishing: An Examination of Phisher Modi Operandi. In *First Usenix Workshop On Large-Scale Exploits and Emergent Threats (LEET)*, 2008.
- [17] T. Moore and R. Clayton. Examining the Impact of Website Take-down on Phishing. In *APWG eCrime Researchers Summit*, Pittsburgh, PA, 2007.
- [18] Netcraft Anti-Phishing Toolbar. <http://toolbar.netcraft.com/>.
- [19] Operated by OpenDNS. Phishtank. <http://www.phishtank.com/>.
- [20] PCWorld. Phishing Scam Takes Aim at Myspace.com. <http://www.pcworld.com/article/id,125956-page,1/article.html?RSS=RSS>, 2006.
- [21] Sophos Web Security and Control Appliance. <http://www.sophos.com/products/enterprise/web/security-and-control/>.
- [22] SpoofStick. <http://www.spoofstick.com/>.
- [23] M. Wu, R. C. Miller, and G. Little. Web Wallet: Preventing Phishing Attacks by Revealing User Intentions. Symposium on Usable Privacy and Security, July 2006.
- [24] Ironport URL Filters. [http://www.ironport.com/technology/ironport\\_url\\_filters.html](http://www.ironport.com/technology/ironport_url_filters.html).
- [25] Y. Zhang, S. Egelman, L. Cranor, and J. Hong. Phishing phish: Evaluating anti-phishing tools. In *Proceedings of the 14th Annual Network & Distributed System Security Symposium (NDSS 2007)*, Feb. 2007.

<sup>3</sup>We assume that active servers hosting detected phishing URLs for the first day alone approximates the trend for such servers appearing later in the week. Figure 3 shows a lower median activity period for servers on the first day because for later days in the trace we have smaller windows of time to estimate activity periods.